

Git & GitHub

국립한경대학교
ICT로봇기계공학부
최현호
hhchoi@hknu.ac.kr

목차

1. Git
2. Github
3. Git 명령어
4. Git 실습
 - 가져오기
 - 업로드
 - 협업하기

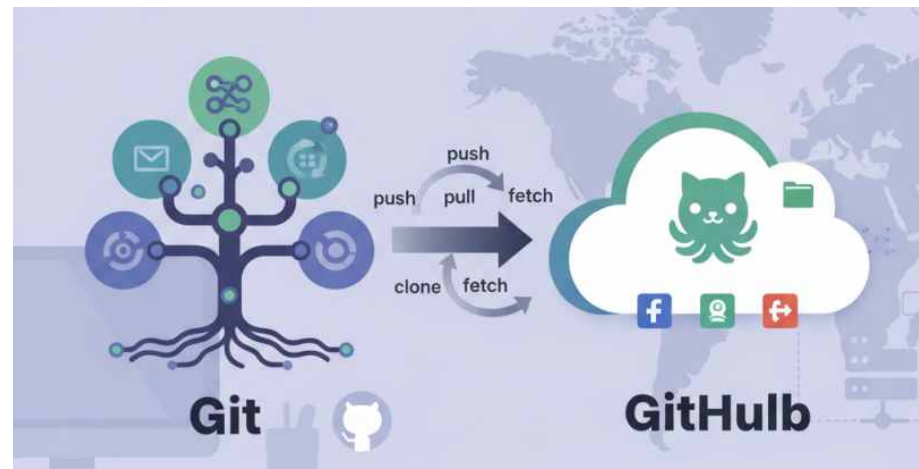
Git 이란?

▪ Git – 버전 관리 도구

- 2005년 리누스 토르발스가 만든 오픈소스 버전 관리 시스템
- 코드의 변경 이력(버전)을 체계적으로 관리하는 도구
- 파일이 언제, 어떻게 수정되었는지 추적하고 이전 상태로 되돌릴 수 있음
- 여러 사람이 동시에 협업하며 충돌 없이 프로젝트를 관리할 수 있음

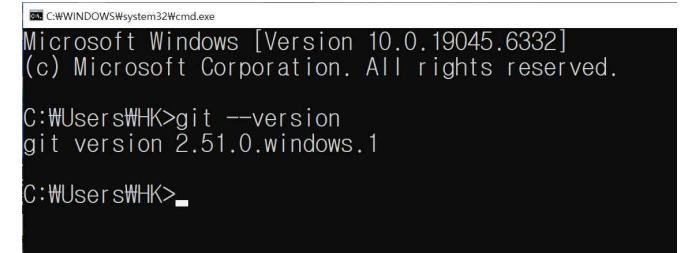
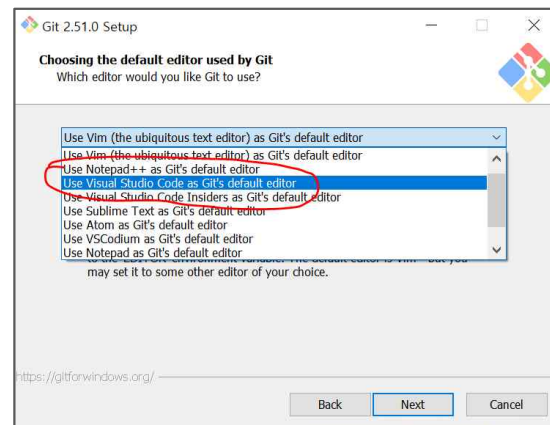
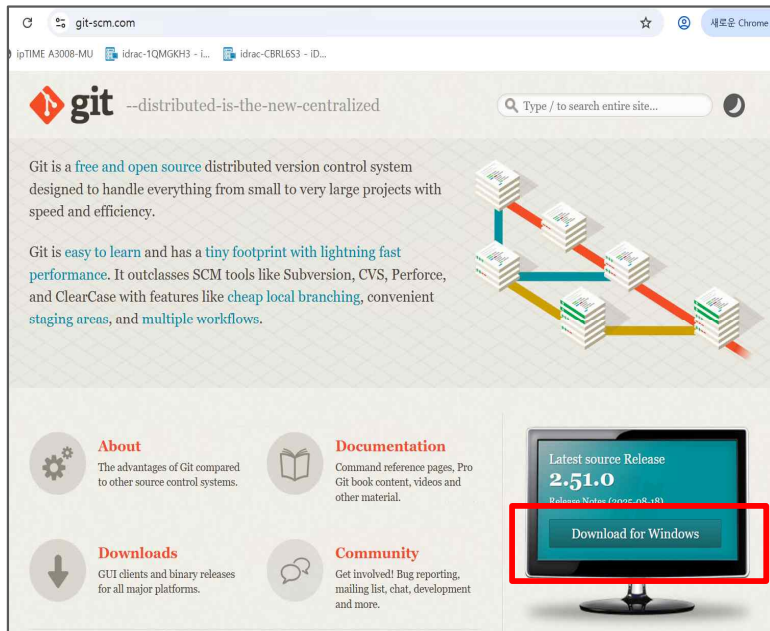
▪ GitHub – 클라우드 협업 플랫폼

- Git으로 관리한 프로젝트를 온라인 저장소(리포지토리) 형태로 보관
- 전 세계 어디서든 접근 가능하고, 팀원들과 공유·협업·리뷰 가능
- 오픈소스 프로젝트 참여 및 포트폴리오 관리에도 활용
- 개발자들의 SNS + 클라우드 드라이브



Git 설치

- 웹 브라우저에서 <https://git-scm.com> 접속
- 메인 화면에서 “Download for Windows” 버튼 클릭
- 다운로드한 Git-2.51.0-64-bit.exe 파일을 더블 클릭하여 실행
- 한가지 제외하고 기본 설정 그대로 설치
 - Editor 선택: **Visual Studio Code** 선택
 - 이외에는 예, Next 또는 Finish 만 누르면 됨
- 설치 확인
 - 명령 프롬프트에서 **git --version** 입력하여 설치된 git 버전 확인:



■ Git 명령어들: git --help를 쳐서 확인

```
C:\WINDOWS\system32\cmd.exe
C:\Users\WHK>git --help
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
          [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
          [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--no-lazy-fetch]
          [--no-optional-locks] [--no-advice] [--bare] [--git-dir=<path>]
          [--work-tree=<path>] [--namespace=<name>] [--config-env=<name>=<envvar>]
          <command> [<args>]

These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
    clone      Clone a repository into a new directory
    init       Create an empty Git repository or reinitialize an existing one


work on the current change (see also: git help everyday)
    add        Add file contents to the index
    mv         Move or rename a file, a directory, or a symlink
    restore    Restore working tree files
    rm         Remove files from the working tree and from the index


examine the history and state (see also: git help revisions)
    bisect     Use binary search to find the commit that introduced a bug
    diff       Show changes between commits, commit and working tree, etc
    grep       Print lines matching a pattern
    log        Show commit logs
    show       Show various types of objects
    status     Show the working tree status


grow, mark and tweak your common history
    backfill   Download missing objects in a partial clone
    branch     List, create, or delete branches
    commit     Record changes to the repository
    merge      Join two or more development histories together
    rebase     Reapply commits on top of another base tip
    reset      Reset current HEAD to the specified state
    switch     Switch branches
    tag        Create, list, delete or verify a tag object signed with GPG


collaborate (see also: git help workflows)
    fetch      Download objects and refs from another repository
    pull       Fetch from and integrate with another repository or a local branch
    push       Update remote refs along with associated objects

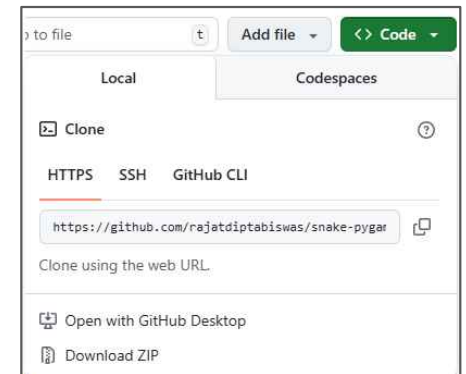
'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
```

git clone

- GitHub에 있는 원격 저장소(프로젝트)를 내 컴퓨터로 복사(clone)

- 실습

- VS code에서 폴더 새로 하나 만들어 열기 (이는 git 프로젝트를 복사시 상위 폴더가 됨)
- 터미널 열고 가져오고자 하는 git 저장소를 복제
 - 예: <https://github.com/rajatdiptabiswas/snake-pygame>
 - `git clone` 뒤에 GitHub 저장소 URL을 붙여 실행
 - 프로젝트 이름(Hello-World)과 동일한 폴더가 자동 생성됨



```
문제  출력  디버그 콘솔  터미널
PS C:\Users\HK\Project> git clone https://github.com/rajatdiptabiswas/snake-pygame.git
Cloning into 'snake-pygame'...
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 24 (delta 6), reused 6 (delta 6), pack-reused 14 (from 1)
Receiving objects: 100% (24/24), 7.50 KiB | 853.00 KiB/s, done.
Resolving deltas: 100% (6/6), done.
PS C:\Users\HK\Project> 
```

■ 실습

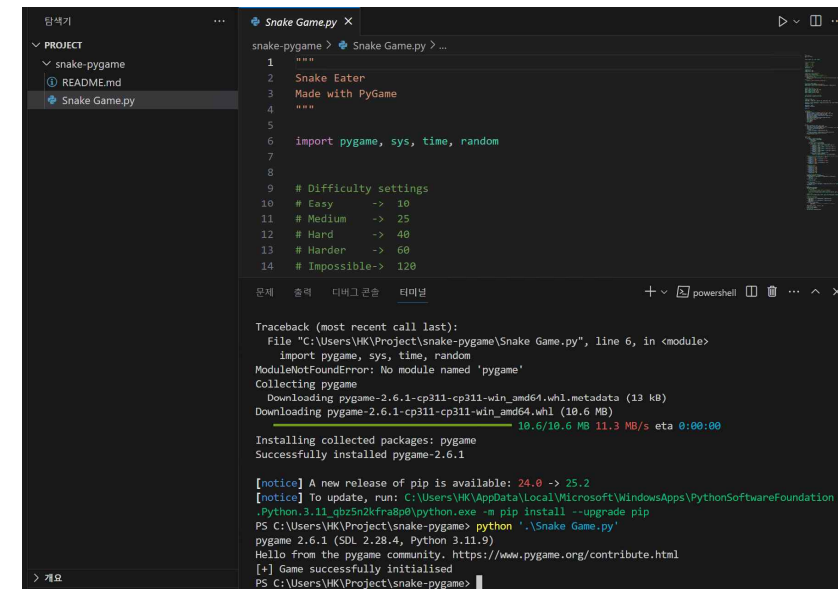
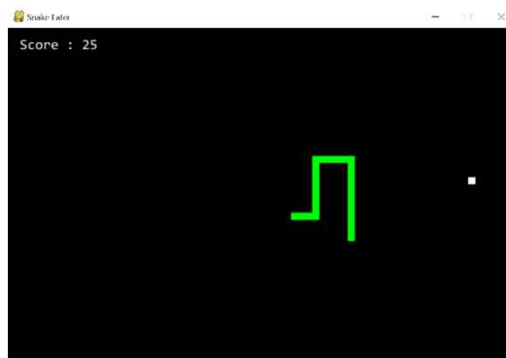
- 폴더 이동: `cd snake-pygame`
- 복제 확인: `git status`

```
PS C:\Users\HK\Project> cd .\snake-pygame\  
PS C:\Users\HK\Project\snake-pygame> git status  
On branch master  
Your branch is up to date with 'origin/master'.  
  
nothing to commit, working tree clean  
PS C:\Users\HK\Project\snake-pygame>
```

→ 지금 master 브랜치에서 작업 중
→ 내 로컬 저장소의 master 브랜치와 원격 저장소 (origin/master)의 상태가 동일하다는 뜻
→ 새로 받아올(git pull) 것도 없고, 새로 올릴(git push) 것도 없는 상태임

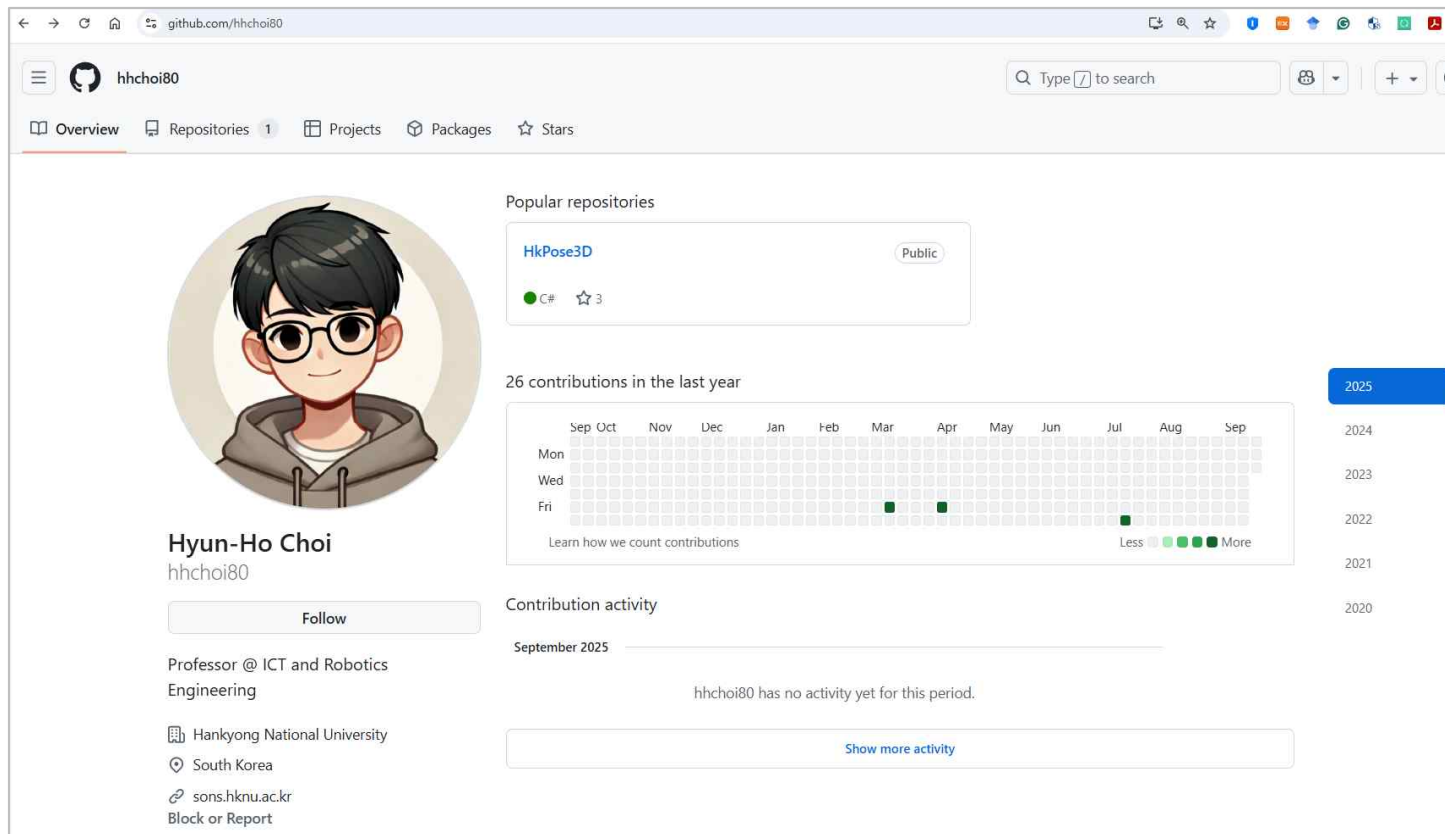
• 실행

- 코드 확인: ls 또는 탐색기 창
- `pip install pygame`
- `python Snake Game.py`



GitHub 가입하기

- 참고: <https://velog.io/@noyohanx/GIT-Github-가입하기>
- 가입 순서
 - GitHub 웹사이트 접속 (<https://github.com>)
 - 회원가입: 페이지 오른쪽 상단 "Sign up" 버튼 클릭
 - 정보 입력
 - 계정 확인: 이메일 주소로 전송된 인증 코드를 입력하여 계정을 확인

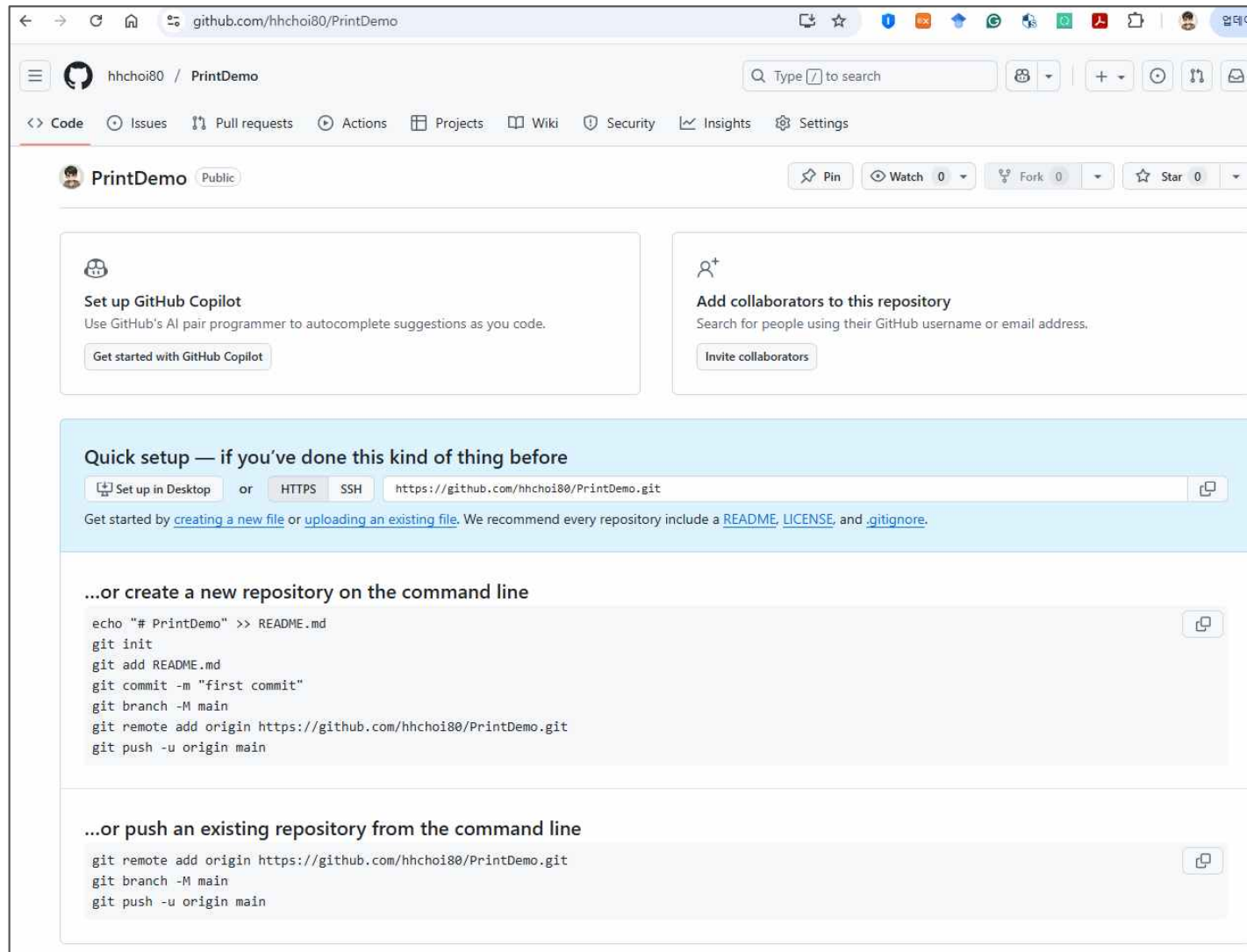


GitHub에서 repository 추가

- GitHub 로그인 > Repositories 메뉴 클릭
- 오른쪽 위 초록색 **New repository** 클릭
- 리포지토리 정보 입력
 - Repository name: 리포지토리 이름 입력 (예: PrintDemo)
 - Description: 리포지토리에 대한 설명을 추가 (옵션)
 - Public/Private: 리포지토리가 공개/비공개 선택 → 일단 public으로 선택
 - Add README: off로 놔둠 → 프로젝트에서 직접 생성
 - Add .gitignore: No. gitignore로 놔둠 → 프로젝트에서 직접 생성
 - Add license: No license로 놔둠
- Create repository 버튼 클릭하여 생성

The screenshot shows the 'Create a new repository' page on GitHub. It is divided into two sections: '1 General' and '2 Configuration'. In the 'General' section, the 'Owner' is 'hhchoi80' and the 'Repository name' is 'PrintExamples', with a green checkmark indicating it is available. The 'Description' field contains 'Python 강의에 사용한 Print 예제'. In the 'Configuration' section, 'Choose visibility' is set to 'Public', 'Add README' is turned 'Off', 'Add .gitignore' is set to 'No .gitignore', and 'Add license' is set to 'No license'. A green 'Create repository' button is at the bottom right.

■ PrintDemo 리포지토리 생성 후 초기 화면



PrintDemo 프로젝트를 github에 저장

- 지난 실습 처럼 PrintDemo 폴더를 만들고 main_print_v1.py 파일 만들기
- 터미널을 열고 다음 명령어들을 순차적으로 수행

1) `git config --global user.name "Hyun-Ho Choi"`

2) `git config --global user.email "hhchoi80@gmail.com"`

- Git을 처음 사용할 때 사용자 정보 설정 필요 → 각자의 계정 정보 (아이디,비번) 넣어서 github로 부터 인증 받아야

3) `git init`

- 현재 폴더를 Git 저장소로 초기화, .git 폴더가 생기면서 버전 관리가 가능해짐

4) `git add .`

- 현재 폴더 안의 모든 변경된 파일을 스테이징(staging)
- 커밋(버전 기록)을 만들기 전에 "이 파일들을 포함시켜라" 하고 준비하는 단계
- .는 "모든 파일"을 의미함 (특정 파일 만 쓰면 그것만 추가 됨)

5) `git commit -m "first commit"`

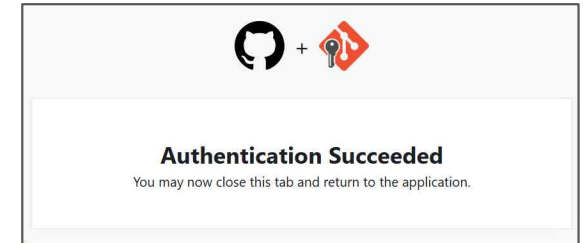
- 스테이징한 파일을 커밋(버전 기록)
- "first commit"은 버전 이름(메시지)이며 자유롭게 작성 가능
- 이 순간이 프로젝트의 첫 버전이 만들어지는 시점

6) `git remote add origin https://github.com/hhchoi80/PrintDemo.git`

- 내 로컬 저장소와 GitHub 원격 저장소를 연결
- origin은 원격 저장소의 별칭 (보통 origin이라고 이름 붙임)
- 뒤 주소는 GitHub에서 만든 저장소 URL

7) `git push -u origin master`

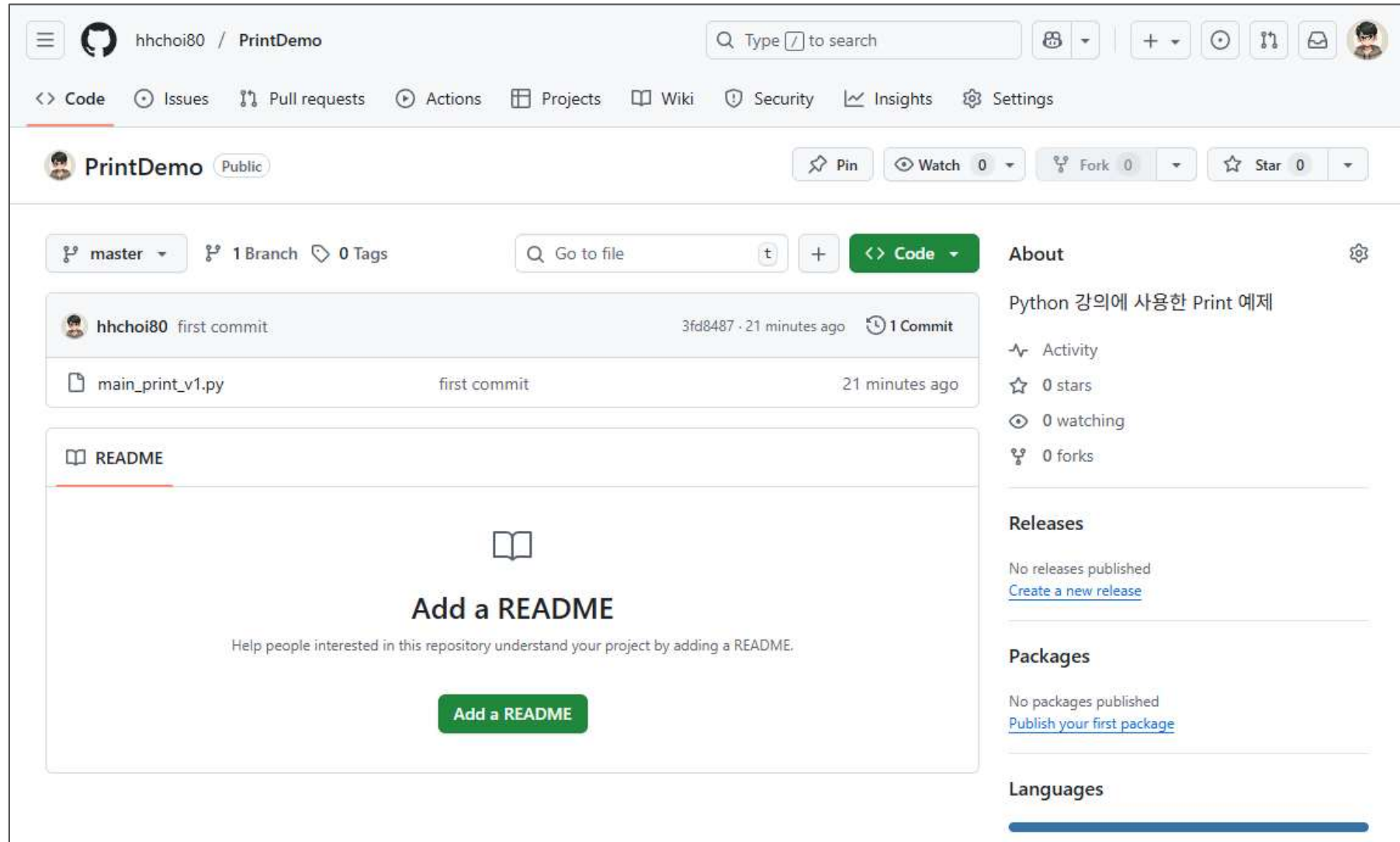
- 내 로컬 커밋을 GitHub 저장소로 업로드
- origin master는 "origin(원격 저장소)"의 "master(브랜치)"로 푸시한다는 뜻
- -u는 앞으로 `git push`만 입력해도 자동으로 이 경로를 사용하도록 설정함



```
문제  출력  디버그 콘솔  터미널
PS C:\Users\HK\Project\PrintDemo> git config --global user.name "Hyun-Ho Choi"
PS C:\Users\HK\Project\PrintDemo> git config --global user.email "hhchoi80@gmail.com"
PS C:\Users\HK\Project\PrintDemo> git init
Reinitialized existing Git repository in C:/Users/HK/Project/PrintDemo/.git/
PS C:\Users\HK\Project\PrintDemo> git add .
PS C:\Users\HK\Project\PrintDemo> git commit -m "first commit"
[master (root-commit) 3fd8487] first commit
1 file changed, 47 insertions(+)
create mode 100644 main_print_v1.py
PS C:\Users\HK\Project\PrintDemo> git remote add origin https://github.com/hhchoi80/PrintDemo.git
PS C:\Users\HK\Project\PrintDemo> git push -u origin master
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 941 bytes | 941.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/hhchoi80/PrintDemo.git
 * [new branch] master -> master
branch 'master' set up to track 'origin/master'.
PS C:\Users\HK\Project\PrintDemo>
```

GitHub Repository 화면

- 파일명, 메시지, 시계모양(history) 등을 클릭해보면 내용 및 상황을 알 수 있음



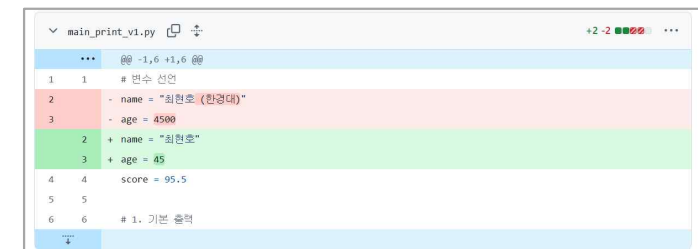
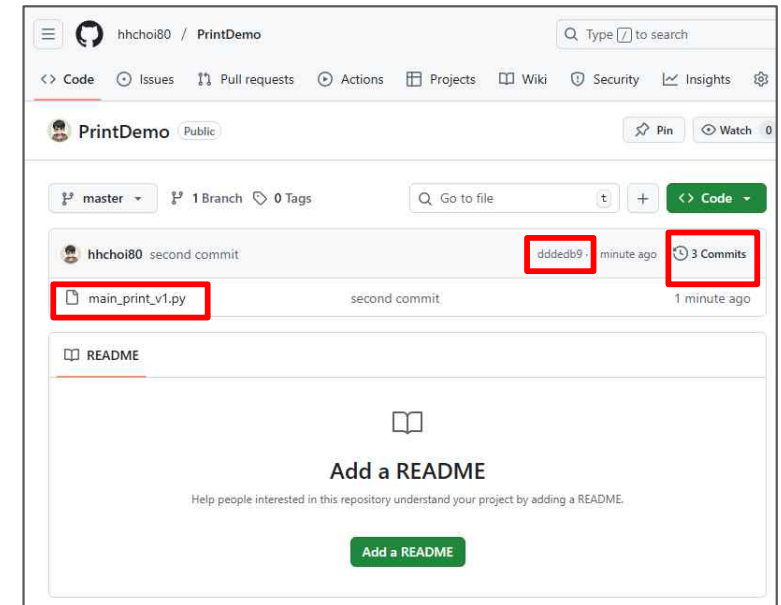
파일 내용 변경 후 다시 업로드

- 파일 내용을 임의로 변경해보기
- 다음 3개의 명령어를 사용해 변경사항 다시 업로드

```
main_print_v1.py > ...
1 # 변수 선언
2 name = "Hyun-Ho Choi"
3 age = 45
4 score = 95.5
5
6 # 1. 기본 출력
7 print("Hello, Python!")
8
9 # 2. 여러 값 출력 (콤마로 구분 → 자동 띄어쓰기)
10 print("Name:", name, "Age:", age, "Score:", score)
11
12 # 3. f-string (가장 많이 쓰임, Python 3.6+)
13 print(f"My name is {name}, I am {age} years old, score: {score}")
14
15 # 4. format() 함수
16 print("My name is {}, I am {} years old, score: {}".format(name, age, score))
```

문제 출력 디버그 콘솔 터미널 + v powershell

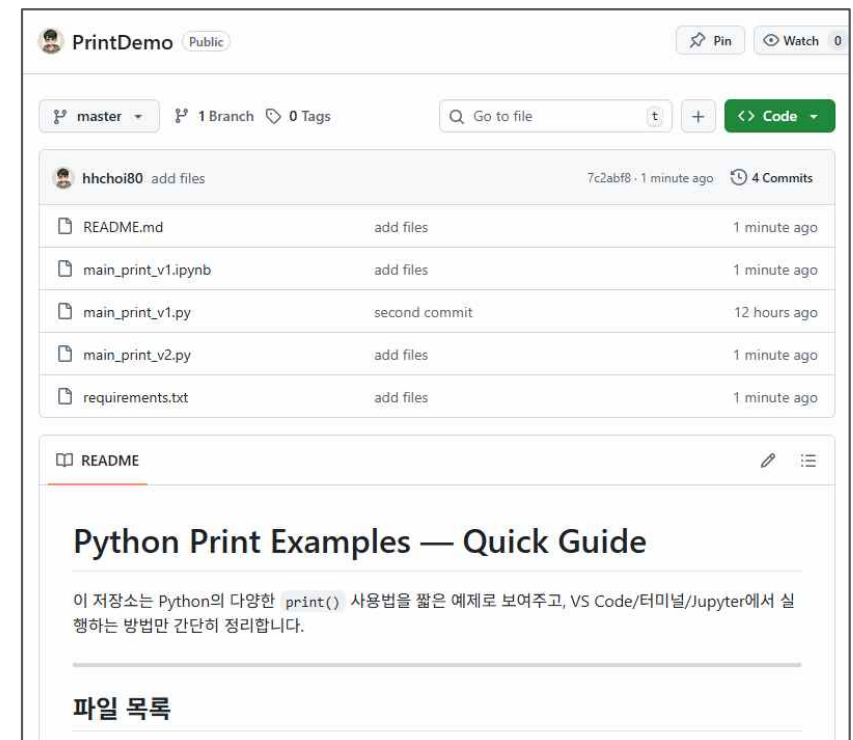
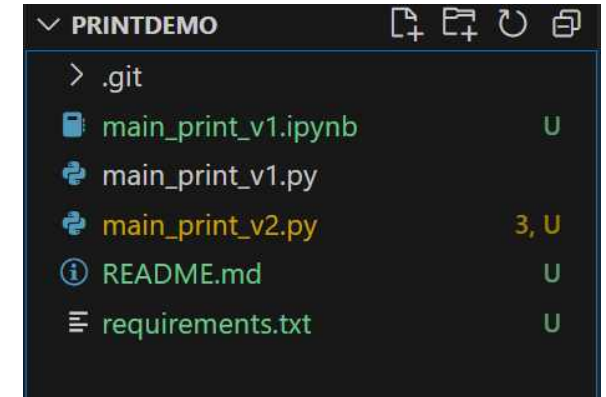
```
PS C:\Users\HK\Project\PrintDemo> git add .
PS C:\Users\HK\Project\PrintDemo> git commit -m "second commit"
[master dddedb9] second commit
1 file changed, 5 deletions(-)
delete mode 100644 .vscode/settings.json
PS C:\Users\HK\Project\PrintDemo> git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (1/1), done.
Writing objects: 100% (2/2), 242 bytes | 242.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/hhchoi80/PrintDemo.git
1223650..dddedb9 master -> master
PS C:\Users\HK\Project\PrintDemo>
```



파일 추가 후 다시 업로드

- README.md 파일 추가
- 다른 파일들 추가
 - 수정 또는 추가된 파일은 다른 색과 U 등이 표시됨
- 다시 업로드를 위해 다음 명령어 순차적으로 수행
 - 1) `git add .`
 - 2) `git commit -m "add files"`
 - 3) `git push`

```
PS C:\Users\HK\Project\PrintDemo> git add .
warning: in the working copy of 'main_print_v1.ipynb', LF will be
ches it
PS C:\Users\HK\Project\PrintDemo> git commit -m "add files"
[master 7c2abf8] add files
4 files changed, 217 insertions(+)
create mode 100644 README.md
create mode 100644 main_print_v1.ipynb
create mode 100644 main_print_v2.py
create mode 100644 requirements.txt
PS C:\Users\HK\Project\PrintDemo> git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 3.23 KiB | 1.62 MiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/hhchoi80/PrintDemo.git
   dddedb9..7c2abf8 master -> master
PS C:\Users\HK\Project\PrintDemo>
```

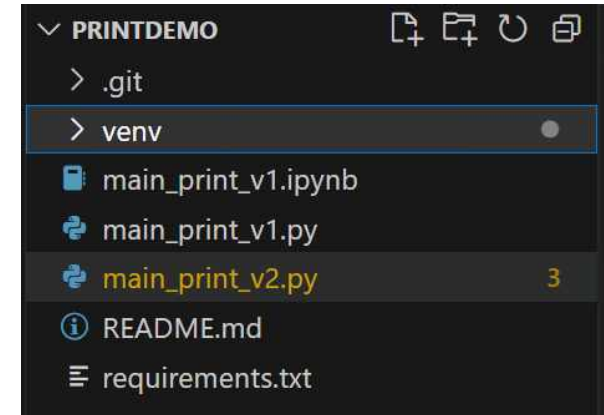


→ README.md 내용이 보임

.gitignore 추가

- 가상환경을 만들고 활성화하기

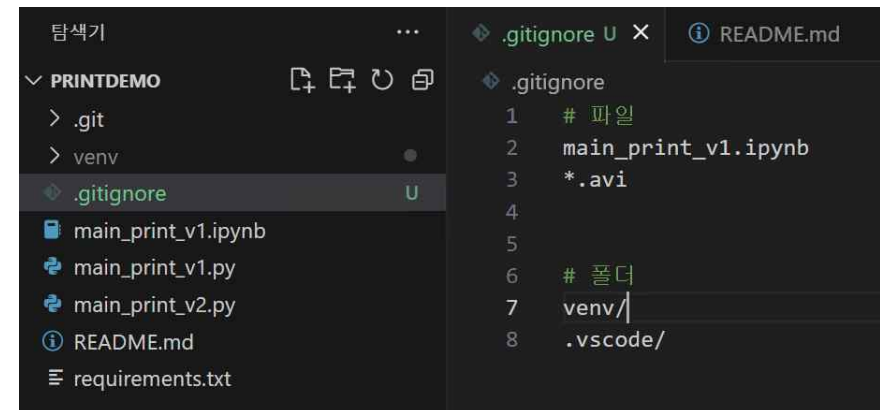
- 1) `python -m venv venv`
- 2) `Set-ExecutionPolicy RemoteSigned -Scope CurrentUser`
- 3) `.\venv\Scripts\activate`



- venv 폴더는 용량이 크고 프로젝트별 맞춤 폴더이므로 github에 올리면 안됨
 - github는 100MB 이내 파일만 업로드 허용

- .gitignore 파일을 만들어 올리지 않을 폴더와 파일 설정해야

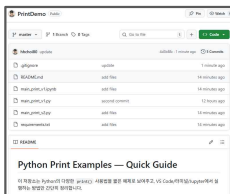
- .gitignore 파일을 생성 (. 포함 스펠링 정확히)
- 파일명, 폴더명을 라인별 입력
- 폴더명에는 끝에 / 넣어줘야
- 파일명에 *는 모든 것을 포함 의미
- 저장



- 다시 업로드

1) `git add .; git commit -m "update"; git push` ← 한줄로 입력시 (;를 사이에 넣으면 됨)

→ venv 폴더는 업로드 안되고, main_print_v1.ipynb 파일은 사라짐, .gitignore 파일은 업로드 됨



Commit 기록 보기

- 지금까지의 Commit 기록을 보려면?
- 다른 옵션들

- git log

```
(venv) PS C:\Users\HK\Project\PrintDemo> git log
commit b5f68ee6119b8e9d70591960724abc340bb3a581 (HEAD -> master, origin/master)
Author: Hyun-Ho Choi <hhchoi80@gmail.com>
Date: Mon Sep 29 06:57:28 2025 +0900

    이름 수정

commit 4d3b88cf48ab0afe804b16dc27ffdbedfc36f807
Author: Hyun-Ho Choi <hhchoi80@gmail.com>
Date: Mon Sep 29 06:53:22 2025 +0900

    update

commit 7c2abf87dbef10d323bf77f58d3484a4f25c5372
Author: Hyun-Ho Choi <hhchoi80@gmail.com>
Date: Mon Sep 29 06:40:28 2025 +0900

    add files

commit dddedb96220c72f4598e5c3833aa3100aa8ace9b
Author: Hyun-Ho Choi <hhchoi80@gmail.com>
Date: Sun Sep 28 18:26:14 2025 +0900

...skipping...
commit b5f68ee6119b8e9d70591960724abc340bb3a581 (HEAD -> master, origin/master)
Author: Hyun-Ho Choi <hhchoi80@gmail.com>
Date: Mon Sep 29 06:57:28 2025 +0900
```

git log -3

git log -p

git log --author="이름"

최근 3개만 보기

코드 변경 내용까지 보기

특정 작성자의 커밋만 보기

- 한 줄 요약 보기

- git log --oneline

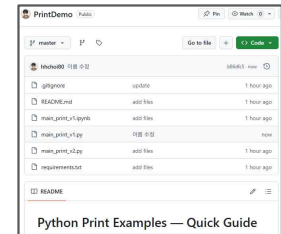
```
(venv) PS C:\Users\HK\Project\PrintDemo> git log --oneline
b5f68ee (HEAD -> master, origin/master) 이름 수정
4d3b88c update
7c2abf8 add files
dddedb9 second commit
1223650 second commit
3fd8487 first commit
```

버전 되돌리기

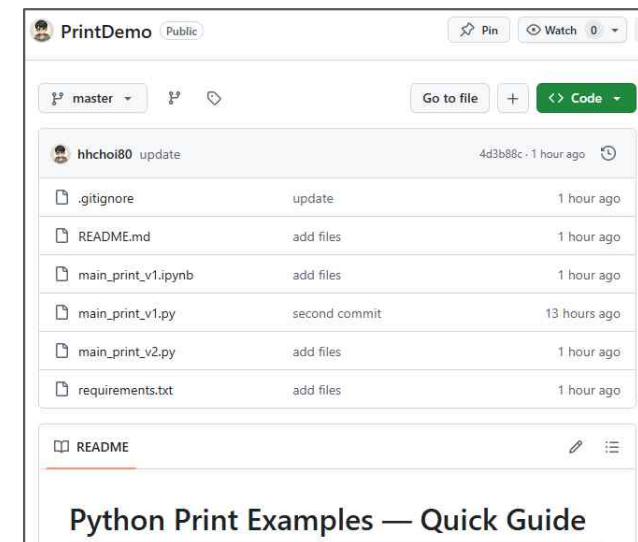
- Main_print_v1.py 파일 임으로 수정 →
- 다시 업로드: `git add .; git commit -m "이름 수정"; git push`
- "push 전 상태(로컬 이전 버전)" 으로만 되돌리고 싶다면
 - > `git reset --hard HEAD~1`
 - HEAD~1 : 한 단계 이전 커밋으로 되돌린다는 뜻
 - --hard : 워킹 디렉토리(실제 코드)까지 완전히 이전 상태로 복원 (즉, add, commit 했던 내용은 모두 사라짐)
 - GitHub에는 아직 이전 버전이 올라가 있음 (즉, 로컬과 원격이 달라짐)
- GitHub 원격 저장소까지 완전히 이전 상태로 되돌리고 싶다면 다음 명령 추가 실행
 - > `git push -f origin master`
 - 위에서 reset으로 되돌린 다음, **강제로 푸시(force push)** 해야
 - 원격 저장소(깃허브)도 이전 버전 상태로 맞춰짐

```
1 # 변수 선언
2 name = "최현호"
```

```
main_print_v1.py > ...
1 # 변수 선언
2 name = "Hyun-Ho Choi"
```



```
(venv) PS C:\Users\HK\Project\PrintDemo> git log --oneline
b5f68ee (HEAD -> master, origin/master) 이름 수정
4d3b88c update
7c2abf8 add files
dddedb9 second commit
1223650 second commit
3fd8487 first commit
(venv) PS C:\Users\HK\Project\PrintDemo> git reset --hard HEAD~1
HEAD is now at 4d3b88c update
(venv) PS C:\Users\HK\Project\PrintDemo> git push -f origin master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/hhchoi80/PrintDemo.git
+ b5f68ee...4d3b88c master -> master (forced update)
(venv) PS C:\Users\HK\Project\PrintDemo>
```



- "커밋은 유지하되 수정만 취소"하고 싶다면









> `git revert HEAD`

- 최근 커밋(HEAD)을 "취소하는 반대 커밋"을 새로 만들
- 기록은 안전하게 남기면서도 코드 상태만 이전으로 되돌아감

```
(venv) PS C:\Users\HK\Project\PrintDemo> git revert HEAD
hint: Waiting for your editor to close the file... █
```

→ 이렇게 나오면 COMMIT_EDITMSG 파일을
닫으면 됨

```
(venv) PS C:\Users\HK\Project\PrintDemo> git revert HEAD
[master 174c584] Revert "이름 수정"
 1 file changed, 1 insertion(+), 1 deletion(-)
(venv) PS C:\Users\HK\Project\PrintDemo>
(venv) PS C:\Users\HK\Project\PrintDemo> git log --oneline
174c584 (HEAD -> master) Revert "이름 수정"
a09a03a (origin/master) 이름 수정
4d3b88c update
7c2abf8 add files
dddedb9 second commit
1223650 second commit
3fd8487 first commit
(venv) PS C:\Users\HK\Project\PrintDemo> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 350 bytes | 350.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/hhchoi80/PrintDemo.git
 a09a03a..174c584 master -> master
(venv) PS C:\Users\HK\Project\PrintDemo> █
```

 PrintDemo Public	
🔗 master ▾ 🔗 1 Branch 🏷 0 Tags 🔍 Go to	
 hhchoi80 Revert "이름 수정" ⋮	
 .gitignore	update
 README.md	add files
 main_print_v1.ipynb	add files
 main_print_v1.py	Revert "이름 수정"
 main_print_v2.py	add files
 requirements.txt	add files

■ 특정 커밋으로 강제로 이동시키는 방법

> `git reset --hard <커밋ID>`

- <커밋ID> : 되돌리고 싶은 커밋의 SHA 해시 (예: a1b2c3d4)
- --hard : 코드와 커밋 이력 모두 해당 시점으로 되돌림
- 아래 예는 "이름 수정" 전 상태 (update 커밋 상황)로 되돌리는 경우

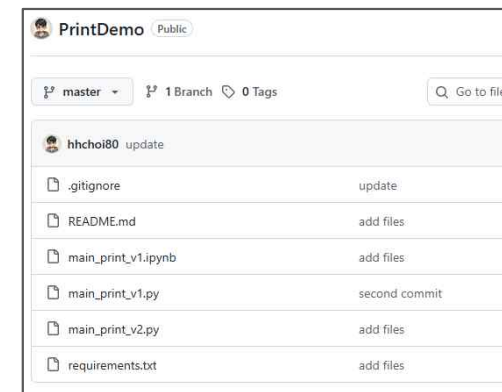
```
(venv) PS C:\Users\HK\Project\PrintDemo> git log --oneline
fd985eb (HEAD -> master) Reapply "이름 수정"
174c584 (origin/master) Revert "이름 수정"
a09a03a 이름 수정
4d3b88c update
7c2abf8 add files
dddedb9 second commit
1223650 second commit
3fd8487 first commit

(venv) PS C:\Users\HK\Project\PrintDemo> git reset --hard 4d3b88c
HEAD is now at 4d3b88c update

(venv) PS C:\Users\HK\Project\PrintDemo> git push
To https://github.com/hhchoi80/PrintDemo.git
! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'https://github.com/hhchoi80/PrintDemo.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. If you want to integrate the remote changes,
hint: use 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

(venv) PS C:\Users\HK\Project\PrintDemo> git push -f origin master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/hhchoi80/PrintDemo.git
+ 174c584...4d3b88c master -> master (forced update)

(venv) PS C:\Users\HK\Project\PrintDemo>
```



← git push하면 에러 발생

← git push -f로 해야

- 특정 커밋 "내용만 취소"하고 이력은 유지하고 싶을 때

> `git revert <커밋ID>`

- <커밋ID>에서 했던 변경만 취소하는 새 커밋이 생성됨
- 기존 이력은 유지됨

```
(venv) PS C:\Users\HK\Project\PrintDemo> git revert 4d3b88c
[master e352a67] Revert "update"
1 file changed, 8 deletions(-)
delete mode 100644 .gitignore
(venv) PS C:\Users\HK\Project\PrintDemo> git log --oneline
e352a67 (HEAD -> master) Revert "update"
4d3b88c (origin/master) update
7c2abf8 add files
dddedb9 second commit
1223650 second commit
3fd8487 first commit
(venv) PS C:\Users\HK\Project\PrintDemo>
```

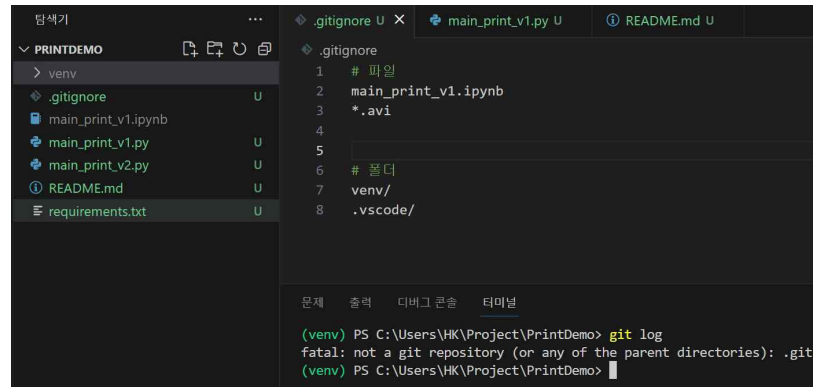
- 특정 커밋 이후 전부 삭제하고 싶을 때

> `git reset --hard HEAD~3`

- 최근 3개의 커밋을 지우고 그 이전 상태로 되돌림
- 특정 커밋을 SHA로 지정하는 것과 결과는 같음

만든 git을 지우려면

- Git은 프로젝트 폴더 안의 .git 폴더에서 모든 버전 관리 정보를 저장
- .git 폴더만 삭제하면 Git도 함께 사라짐
- 코드는 그대로 남지만, 버전 관리 기록·커밋·브랜치 정보는 전부 사라짐



```
.gitignore
# 파일
main_print_v1.ipynb
*.avi

# 폴더
venv/
.vscode/

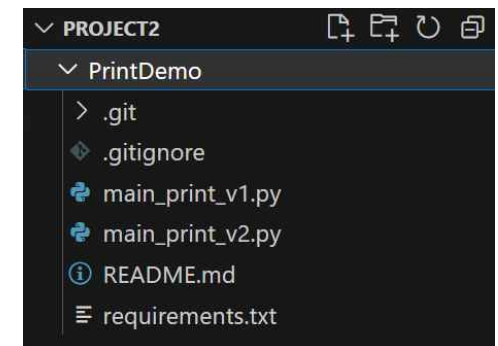
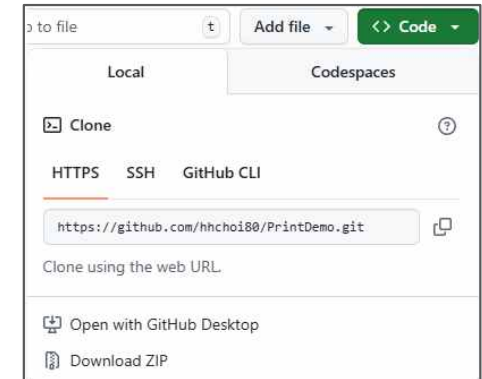
(venv) PS C:\Users\HK\Project\PrintDemo> git log
fatal: not a git repository (or any of the parent directories): .git
(venv) PS C:\Users\HK\Project\PrintDemo>
```

- 삭제 후에 다시 github와 연결하려면 다음 명령어들을 다시 실행하면 됨
 - 1) git init
 - 2) git add .
 - 3) git commit -m "first commit"
 - 4) git remote add origin <https://github.com/hhchoi80/PrintDemo.git>
 - 5) git push -u origin master

다른 곳에서 git pull 하기

- 새 VS Code 열고 (ctrl+shift+N) 새 폴더 열기
 - 예: Project2 이름으로 폴더 만들고 열기
- 터미널에서 만들었던 git repository 복제 하기
 - > `git clone https://github.com/hhchoi80/PrintDemo.git`
 - PrintDemo가 그대로 다운로드 된 것을 확인
- git pull 수행해보기
 - 아래와 같이 새로 만들어진 PrintDemo 폴더로 들어가서 해야
 - git pull을 하면 이미 새것이라고 나옴 (업데이트 된게 없어서)

```
PS C:\Users\HK\Project2> git clone https://github.com/hhchoi80/PrintDemo.git
Cloning into 'PrintDemo'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 7 (delta 0), reused 7 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (7/7), done.
❌ PS C:\Users\HK\Project2> git pull
● fatal: not a git repository (or any of the parent directories): .git
PS C:\Users\HK\Project2> cd .\PrintDemo\
● PS C:\Users\HK\Project2\PrintDemo> git pull
Already up to date.
```



- 원래 프로젝트에서 임의로 파일 내용 수정하고 git push 하기

```
main_print_v1.py > ...  
1  # 변수 선언  
2  name = "최현호-최현호"  
3  age = 45  
4  score = 95.5  
5
```

```
PS C:\Users\HK\Project\PrintDemo> git add .; git commit -m "이름 수정"; git push  
[master 1a0d907] 이름 수정  
1 file changed, 1 insertion(+), 1 deletion(-)  
Enumerating objects: 5, done.  
Counting objects: 100% (5/5), done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (3/3), 323 bytes | 323.00 KiB/s, done.  
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)  
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.  
To https://github.com/hhchoi80/PrintDemo.git  
93e31c6..1a0d907 master -> master  
PS C:\Users\HK\Project\PrintDemo>
```

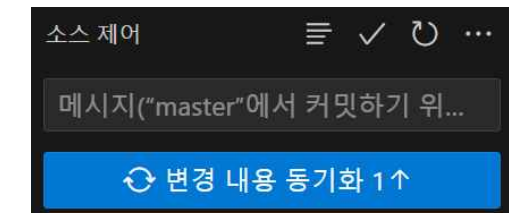
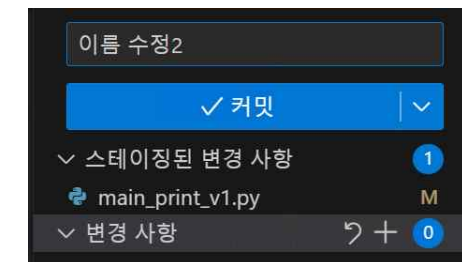
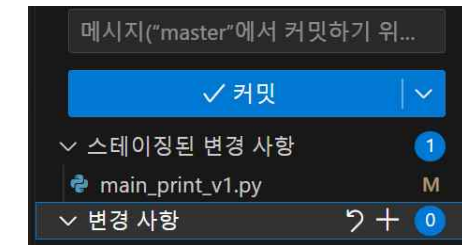
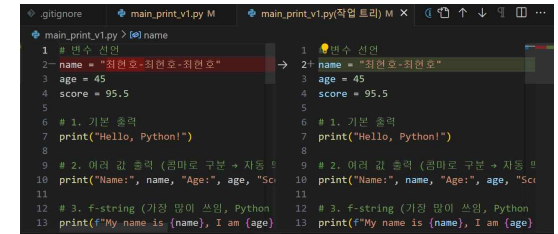
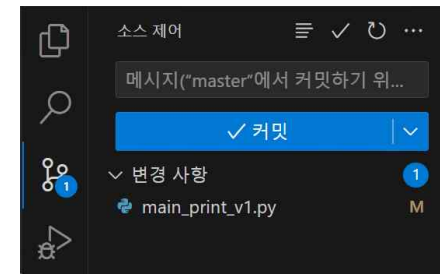
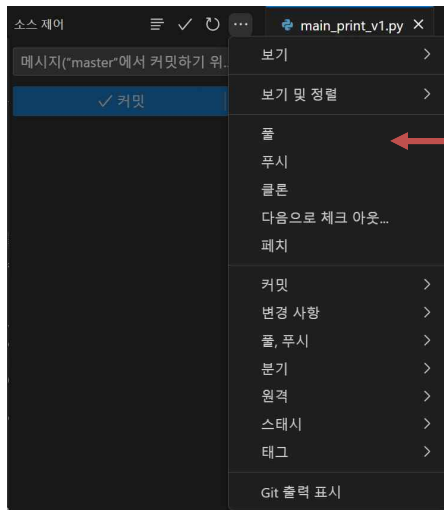
- 이제 새로운 프로젝트에서 git pull 하기
 - 수정 내역이 보이면서, 파일 내용이 변경됨

```
PS C:\Users\HK\Project2\PrintDemo> git pull  
remote: Enumerating objects: 5, done.  
remote: Counting objects: 100% (5/5), done.  
remote: Compressing objects: 100% (1/1), done.  
remote: Total 3 (delta 2), reused 3 (delta 2), pack-reused 0 (from 0)  
Unpacking objects: 100% (3/3), 303 bytes | 33.00 KiB/s, done.  
From https://github.com/hhchoi80/PrintDemo  
93e31c6..1a0d907 master -> origin/master  
Updating 93e31c6..1a0d907  
Fast-forward  
main_print_v1.py | 2 +  
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
PrintDemo > main_print_v1.py > ...  
1  # 변수 선언  
2  name = "최현호-최현호"  
3  age = 45  
4  score = 95.5  
5
```

VS Code에서 쉽게 쓰기

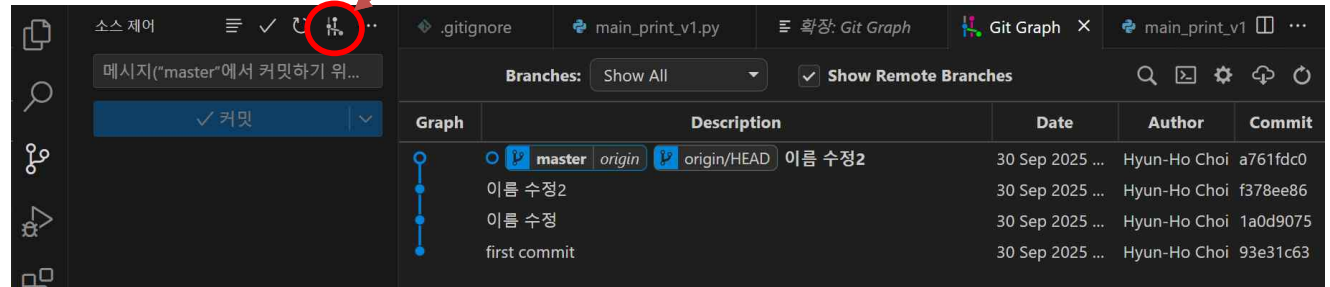
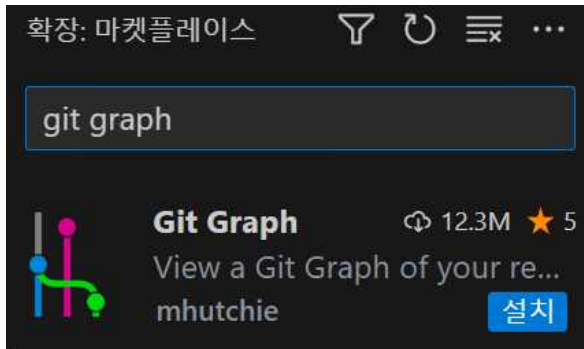
- 왼쪽 깃허브 모양의 메뉴 아이콘 클릭하면
 - 파일명 더블 클릭: 작업트리 열림 (변경사항 확인 가능)
 - + 아이콘 눌러서 변경 내용 스테이징 하기
 - `git add .` 와 동일
 - 커밋 메시지 입력후 커밋 버튼 클릭
 - `git commit -m "메시지"` 와 동일
 - 변경 내용 동기화 클릭
 - `git pull; git push` 와 동일
- 다른 프로젝트에서 `git pull` 할때
 - ... 아이콘 > 풀(Pull) 클릭



Git 관련 유용한 Extensions

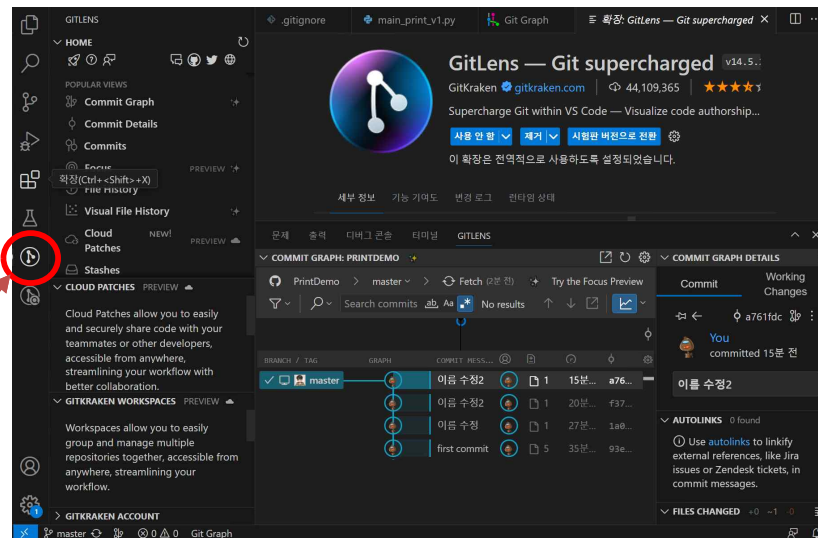
■ Git graph

- 커밋 히스토리를 한눈에 보여줌



■ Gitlens

- 코드와 커밋 이력을 깊이 있게 분석하는 도구



협업하기: Branch 만들고 합치기

■ 브랜치 (개인 작업 공간) 만들기

- 각 브랜치에서 자유롭게 코드를 수정해도 master(또는 main)에는 영향을 주지 않음
- A는 이름을 변경, B는 나이를 변경한다고 가정
- A에서 다음과정 수행

■ 코드에서 이름 변경

- > git branch update-name # 브랜치 생성
- > git checkout update-name # 브랜치 이동
- > git add .
- > git commit -m "이름 수정"
- > git push origin update-name

```
git@hhoi: ~$ git branch update-name
git@hhoi: ~$ git checkout update-name
Switched to branch 'update-name'
git@hhoi: ~$ git add .
git@hhoi: ~$ git commit -m "이름 수정"
[master 0000000] 이름 수정
1 file changed, 1 insertion(+), 1 deletion(-)
git@hhoi: ~$ git push origin update-name
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 322 bytes | 322.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote: Create a pull request for 'update-name' on GitHub by visiting:
https://github.com/hhoi/PrintDemo/pull/new/update-name
To https://github.com/hhoi/PrintDemo.git
 * [new branch] update-name -> update-name
git@hhoi: ~$
```

• B에서 다음과정 수행

■ 코드에서 나이 변경

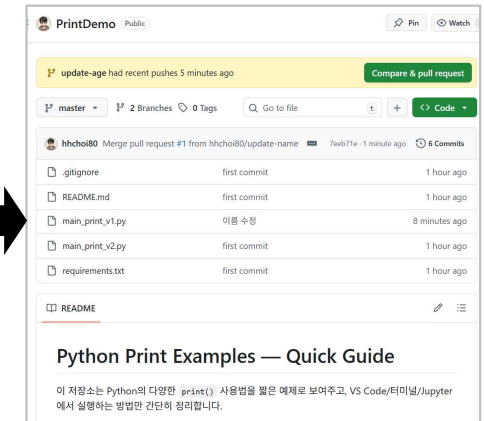
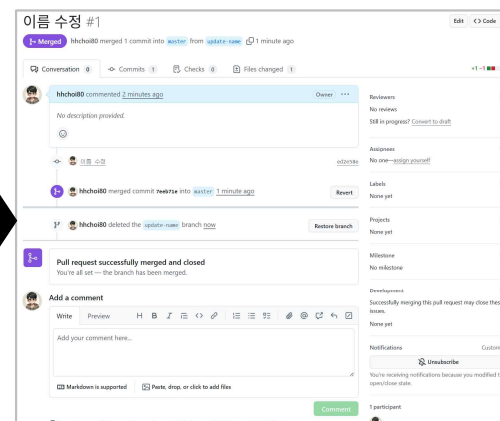
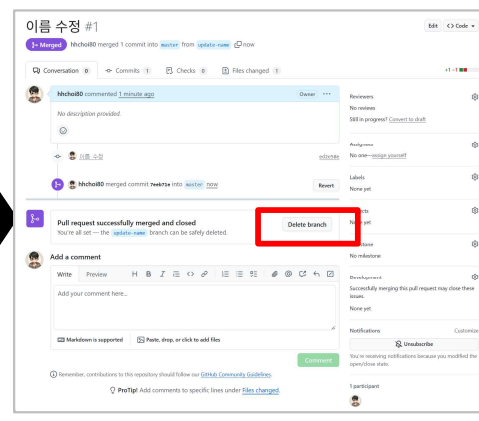
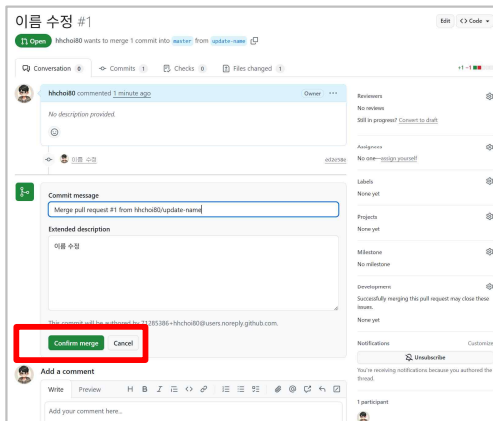
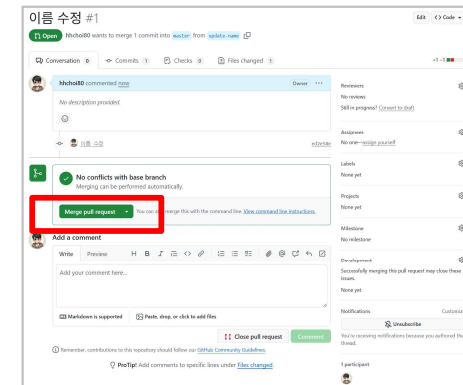
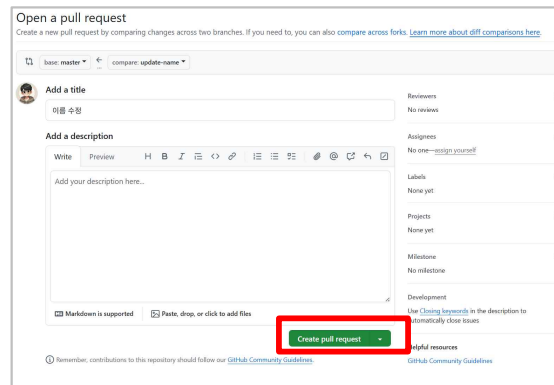
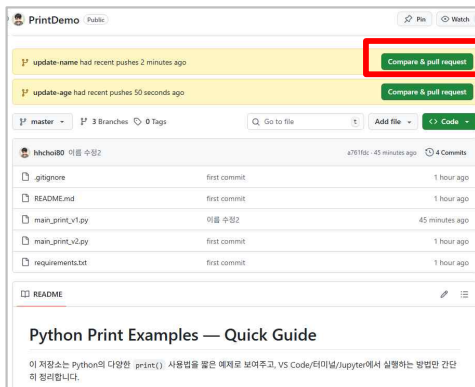
- > git branch update-age # 브랜치 생성
- > git checkout update-age # 브랜치 이동
- > git add .
- > git commit -m "나이 수정"
- > git push origin update-age

```
PrintDemo > git branch update-age
PrintDemo > git checkout update-age
Switched to branch 'update-age'
PrintDemo > git add .
PrintDemo > git commit -m "나이 수정"
[update-age 53ec325] 나이 수정
1 file changed, 1 insertion(+), 1 deletion(-)
PrintDemo > git push origin update-age
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 310 bytes | 310.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote: Create a pull request for 'update-age' on GitHub by visiting:
https://github.com/hhoi/PrintDemo/pull/new/update-age
To https://github.com/hhoi/PrintDemo.git
 * [new branch] update-age -> update-age
PrintDemo >
```

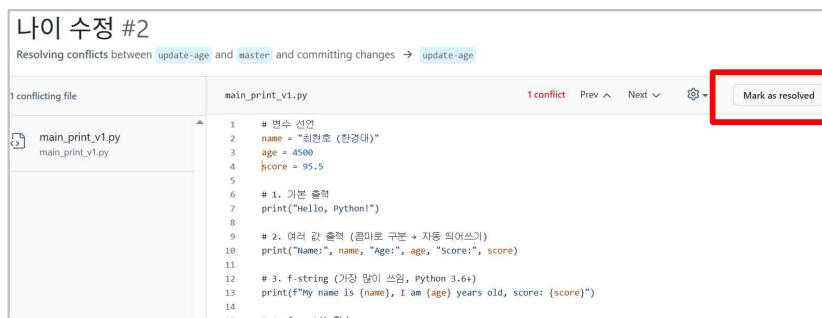
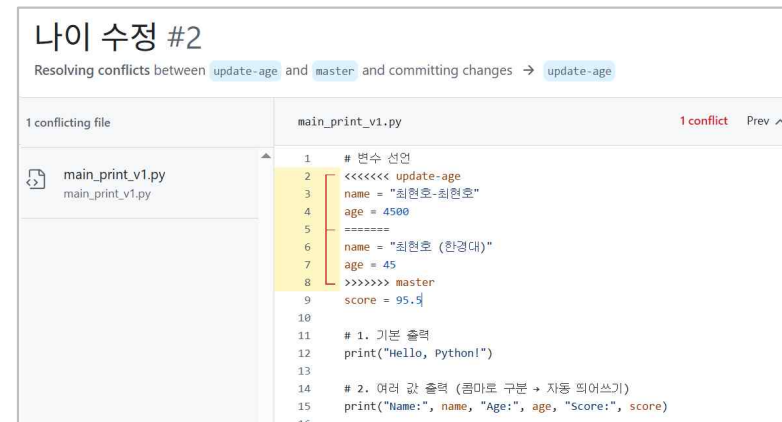
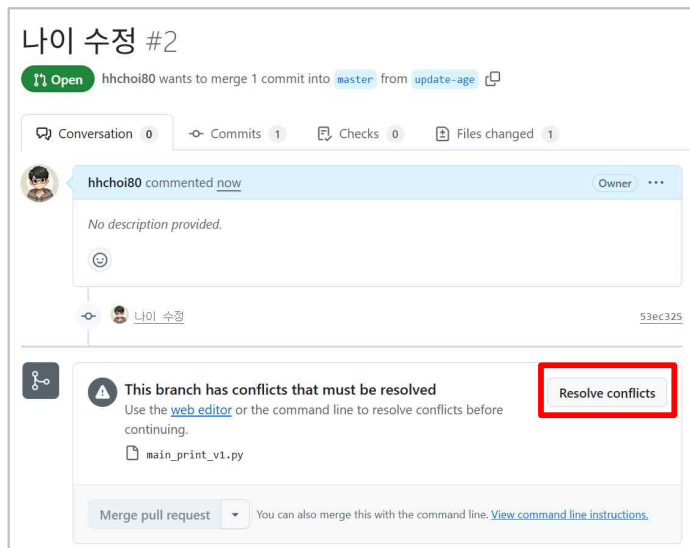
- GitHub에 login-feature와 signup-feature라는 두 개의 브랜치가 생김

협업하기: Branch 만들고 합치기

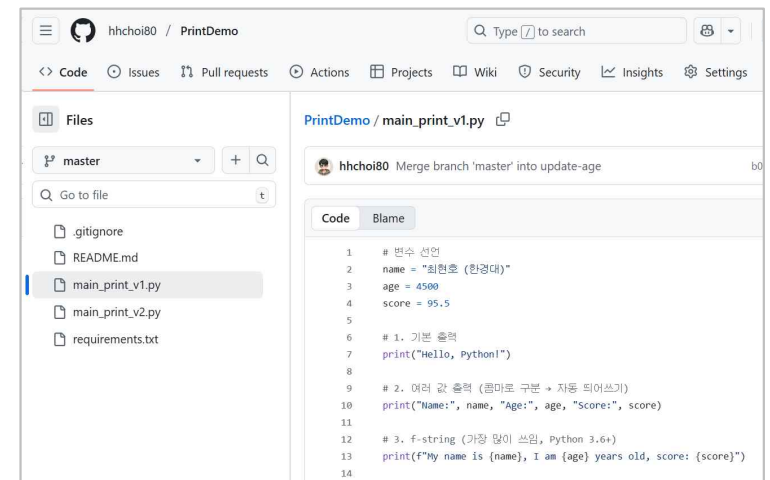
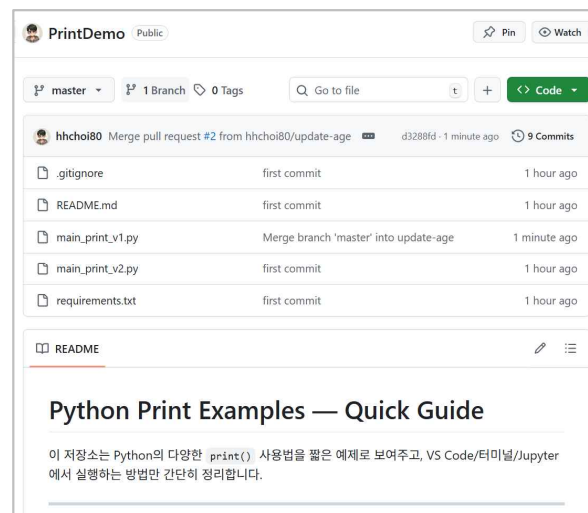
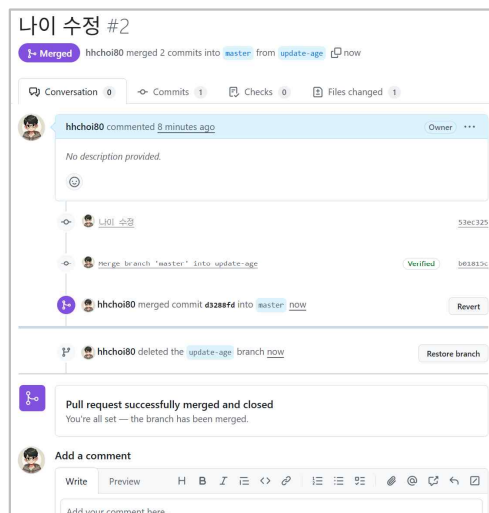
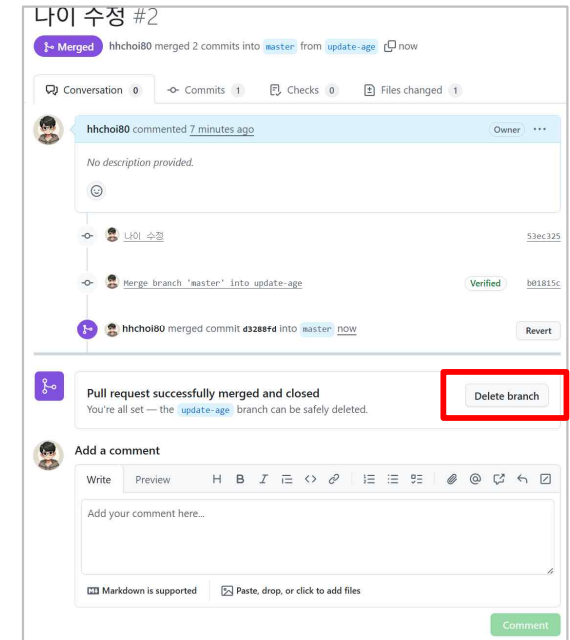
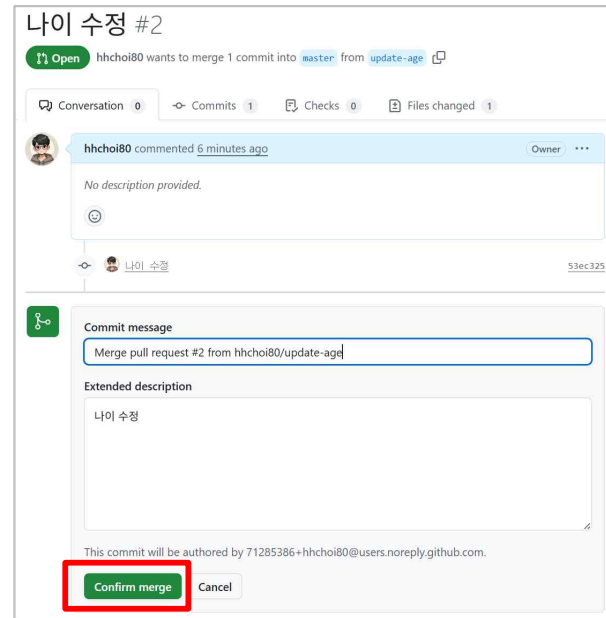
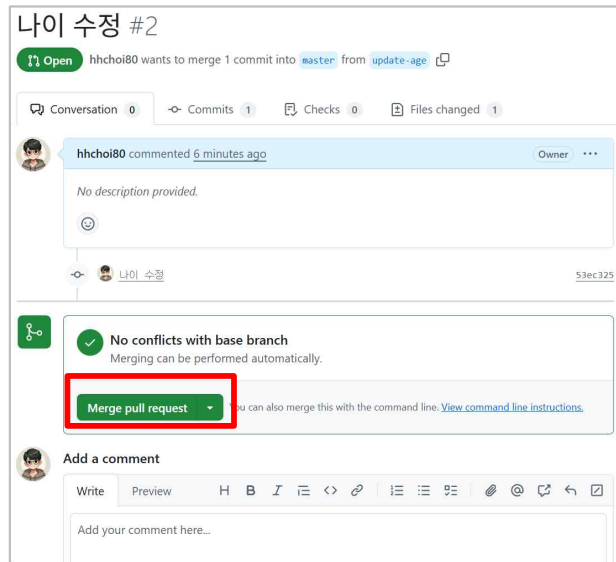
- GitHub에서 Pull Request 만들기 (실무에서 가장 많이 사용)
 - Pull Request (PR): 내가 만든 브랜치의 코드를 master 브랜치에 합쳐도 될지를 요청하는 것
 - GitHub 페이지에서 "Compare & pull request" 클릭
 - 리뷰 → Merge 버튼 클릭
 - 완료 후 브랜치는 필요 없으면 삭제 가능



- 두번째 branch에 대해서도 “Compare & pull request” 클릭
 - Conflict (충돌) 발생시 수동으로 해결 (웹에서 코드 편집)

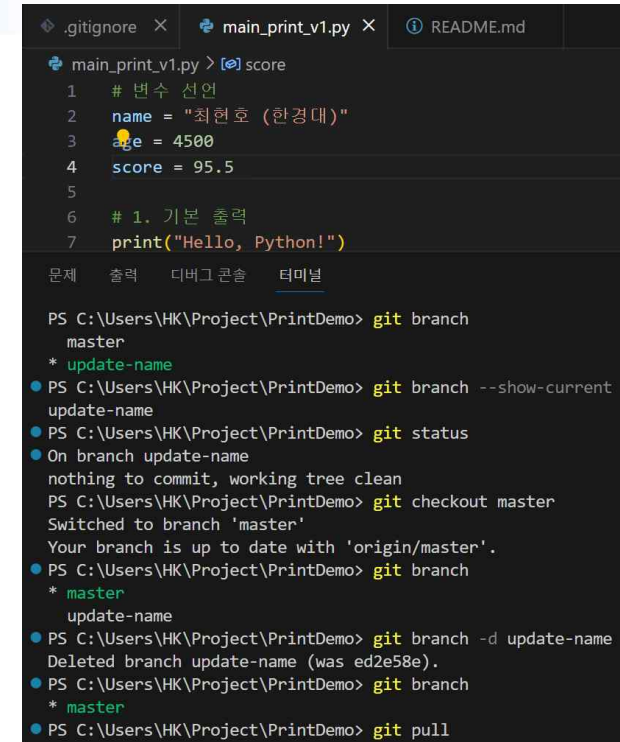


■ 이후 과정은 동일



로컬에서 branch 관리

- 현재 내가 속한 branch 확인
 - > git branch
 - > git branch --show-current
 - > git status
- Master 브랜치로 이동하기
 - > git checkout master
 - > git switch master
- 로컬 브랜치 삭제하기
 - 병합 완료 후: git branch -d 브랜치이름 (예: git branch -d update-name)
 - 병합 완료 전: git branch -D 브랜치이름 (예: git branch -D update-name)
- Merge 후 다시 받아오기
 - > git pull



The screenshot shows a code editor with a file named `main_print_v1.py` containing Python code. The code defines variables `name` and `age`, and a `score` variable, followed by a `print` statement. Below the code, a terminal window shows a series of Git commands and their outputs. The commands include `git branch`, `git branch --show-current`, `git status`, `git checkout master`, `git branch`, `git branch -d update-name`, and `git pull`. The terminal output shows the current branch is `update-name`, the working tree is clean, and the branch is successfully deleted and pulled.

```
.gitignore X main_print_v1.py X README.md
main_print_v1.py > [e] score
1 # 변수 선언
2 name = "최현호 (한경대)"
3 age = 4500
4 score = 95.5
5
6 # 1. 기본 출력
7 print("Hello, Python!")

문제 출력 디버그 콘솔 터미널

PS C:\Users\HK\Project\PrintDemo> git branch
master
* update-name
PS C:\Users\HK\Project\PrintDemo> git branch --show-current
update-name
PS C:\Users\HK\Project\PrintDemo> git status
On branch update-name
nothing to commit, working tree clean
PS C:\Users\HK\Project\PrintDemo> git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
PS C:\Users\HK\Project\PrintDemo> git branch
* master
update-name
PS C:\Users\HK\Project\PrintDemo> git branch -d update-name
Deleted branch update-name (was ed2e58e).
PS C:\Users\HK\Project\PrintDemo> git branch
* master
PS C:\Users\HK\Project\PrintDemo> git pull
```

기타 유용한 Git 명령어

명령어	기능 / 설명	사용 예시
git diff	최근 변경 사항(수정된 코드)을 비교해서 보여줌	git diff
git diff --staged	스테이징된 변경 내용만 비교	git diff --staged
git show <커밋ID>	특정 커밋의 상세 내용(작성자, 날짜, 변경 코드) 확인	git show a1b2c3d
git log --graph --oneline --decorate --all	브랜치 흐름을 트리 형태로 시각화해서 보여줌	협업 시 브랜치 구조 확인용
git stash	현재 변경사항을 임시 저장 (커밋 없이 잠시 보관)	git stash
git stash list	저장된 stash 목록 보기	git stash list
git stash pop	임시 저장한 변경사항을 다시 적용하고 stash에서 제거	git stash pop
git stash apply	임시 저장한 변경사항을 다시 적용 (stash는 유지)	git stash apply
git tag <태그명>	특정 커밋에 태그(버전 이름) 붙이기	git tag v1.0.0
git tag -d <태그명>	태그 삭제	git tag -d v1.0.0
git checkout <커밋ID>	특정 커밋 시점으로 임시 이동 (detached HEAD 상태)	git checkout a1b2c3d
git reset --soft <커밋ID>	특정 커밋으로 되돌리되, 변경 내용은 그대로 유지	git reset --soft HEAD~1
git reset --mixed <커밋ID>	스테이징만 취소하고 작업 내용은 유지	git reset --mixed HEAD~1
git reset --hard <커밋ID>	특정 커밋으로 완전히 되돌림 (변경 내용 모두 삭제)	git reset --hard HEAD~1
git reflog	과거 HEAD 이동 이력(브랜치 전환, reset 등) 전체 기록 확인	실수 복구 시 유용
git clean -fd	Git이 추적하지 않는(추가만 된) 파일/폴더 삭제	git clean -fd
git blame <파일명>	각 코드 줄이 누가, 언제 수정했는지 확인	git blame main.py
git remote -v	연결된 원격 저장소 주소 확인	git remote -v
git fetch	원격 저장소 변경 사항만 가져오기 (병합은 하지 않음)	git fetch origin
git cherry-pick <커밋ID>	특정 커밋 하나만 현재 브랜치에 가져오기	git cherry-pick a1b2c3d
git revert <커밋ID>	특정 커밋의 변경만 되돌리는 "역커밋" 생성	git revert a1b2c3d
git pull --rebase	최신 코드를 받아올 때 커밋 이력을 깔끔하게 정리하며 병합	협업 시 충돌 줄이기 용도

참고 자료

- 얕팍한 코딩사전 : <https://youtu.be/1I3hMwQU6GU>
- 에러 발생시 검색 및 AI 활용

- 본인 Github에 PrintDemo 저장소에 아래와 같이 파일을 올리고 해당 저장소 주소를 사이버캠퍼스에 적어주세요.
 - 주소 예: <https://github.com/hhchoi80/PrintDemo>
 - 커밋 메시지는 달라도 상관 없음
 - 사캠에 파일은 올릴 필요 없음

